International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume-2, Special Issue 1 (July-Aug 2014), PP. 28-32

ENHANCING THE ADAPTIVITY OF ENCRYPTION FOR STORAGE ELECTRONIC DOCUMENTS

Nour Mahmoud Khafajah , Kamaruzzaman Seman
Faculty of Science and Technology,
University Science Islam Malaysia (USIM)
71800 Nilai, Negari Sembilan, Malaysia.

Osama Ahmed Khashan
Faculty of Information Science and Technology,
National University of Malaysia (UKM)
43600 Bangi, Selangor, Malaysia

***Abstract*— the rapid advancement in the domain of information technology has increased the amount of our sensitive documents stored on disk drives and removable storage media. Although many encryption applications and software protection systems are available to provide trusted protection of those documents, they often fail to pay sufficient attention to the increasing challenges of satisfying security implications on storage domain. This, thus, results in greater chances for security breaches and intrusion attacks, in addition to the greatly increased costs to business and end users. Developing a storage protection system based on involving the reuse approach in every phase of a system development can help in analyzing risks and security policies, identifying threats, and determining security requirements. This paper discusses the value of reusability for specifying security requirements of current storage cryptographic systems. Then, we propose a cryptographic model based on a filter driver technology focuses on protecting storage document files. Such proposed model can be able to resolve obstacles to the security requirements identified, and to meet its goal of a high assurance storage protection system.**

***Index Terms*— Reusability, file system filter driver, transparent encryption, stored documents protection.**

I. INTRODUCTION

Electronic documents touch almost every aspects of our daily life in modern information technology. Many sensitive documents of private users' information and valuable business details are stored on their computers; where the leak of such confidential documents can result in expose user privacy, heavy financial losses, losing revenue, compromised ability to compete, and much more. Therefore, security of electronic documents at storage domain becomes an increasingly growing problem and challenging issue, in addition to the essential need of security expertise to meet non-functional security requirements [1].

The security of data in storage domain is needed for long term and may be in terms of years, unlike communication, where the security is only needed in terms of seconds and once the data reach its target, the job is done. Secure storage software applications nowadays are more and more ubiquitous, but they are so far vulnerable, heterogeneous, and safety-critical. Moreover, the attacks and threats become more and more widespread, sophisticated and malicious, which may be mounted by various kinds of attackers. Thus, this make the data at rest are always at risk due both to various types of threats and the vulnerabilities of protection system that may have [2].

Security requirements engineering is an important part of a system development process to address different software security issues at the early stage of development cycle, and then provides methods and techniques that are able to tackle such specified security issues [3]. The security requirements of any system roughly depend on the environment in which the system is deployed. Furthermore, security requirements tend to be more standardized with their associated security mechanism (i.e. cryptography, authentication, etc.) and their architectural mechanism. Thus, this can help in identifying the optimum way of fulfilling the security requirements [4].

Reuse of security requirements can efficiently help in discovering security issues of existing security systems in order to improve the quality of development processes, inspire new ideas from reusable components, and significantly save development time and cost [5, 6].

Most of storage security systems have been developed with poor understanding of security concerns, where the specification of security requirements are often defined as an incomplete subset and typically not integrated with different phases of the system development. Even if there is an attempt to define security requirements, the developers tend to describe the design solutions using standard security requirements rather than making declarative propositions of real requirements with regard to the required protection level [7]. Therefore, storage protection systems are recognized to be poor-quality software protections, and they are always influenced by the security requirements [8].

Storage encryption still a far more cost-effective solution among other storage protections to ensure the primary security aspects of confidentiality, integrity and availability of storage data in the face of a hostile intruder, if it's properly implemented and appropriately applied [9]. As consequence, a large number of cryptographic developments for guaranteeing the trusted protection of the storage; nevertheless, the security level of a cryptographic scheme may be reasonable from its designer has in mind, but the attack is never anticipated. Thus, in many cases it has been found that the problem is not in the cryptographic design as much as understanding of security requirements for storage encryption.

In this paper, we specify the security-related requirements of cryptographic systems on storage domain. Then, we based on the reuse approach for the specified requirements to propose a storage cryptosystem model that can be able to link the security of the storage documents, the risk environment and the security requirements to securely manage, control, monitor, distribute and to permit the usage of the cryptographic mechanism.

The rest of this paper is organized as follows. Section 2 gives an overview of storage cryptosystem models. Section 3 discusses the problems and challenges with storage encryption. Section 4 describes the security requirements for storage domain cryptographic systems. The proposed model of storage

cryptographic documents is presented in Section 5, and the conclusion is given in section 6.

II. STORAGE CRYPTOSYSTEMS OVERVIEW

A cryptographic scheme in the storage domain shall be a set of hardware, software, or some combination thereof that implements cryptographic functions or processes, including cryptographic algorithms and, parameters, key generation, and is contained within a defined cryptographic boundary.

Software-based cryptosystems are nowadays widely used to provide the encryption utility with several different options. These software-based cryptosystems can be carried out using third party encryption applications that are mounted on the user space to provide generic users with cryptographic service for different storage data types and platforms, such as such as crypt, aescrypt, Cryptainer, BitLocker, etc. These application schemes work by taking the file name of the file that needs to be encrypted and the password as inputs to produce a ciphered version of that file. Therefore, the user is required to supply the key (password) for each file he needs to encrypt and required to remember it in order to retrieve the file back from the disk for read. Some encryption applications have been implemented with built-in cryptographic libraries that are able to provide automated cryptographic functionality when a file's data is written or read, such as a text editor [10].

Storage encryption can be implemented as a basic part of the operating system file systems to perform all cryptographic and key management operations. The encryption mechanism is carried out at different location layers to encrypt or decrypt the whole single or multiple disk partitions, or to encrypt individual files or directories. It can be performed as a user space encryption layer using FUSE (Filesystem in Userspace) technology [11] for Linux platform. It can also be operated as a middleware layer inside the kernel space by inserting a cryptographic file system layer in the level between a user space and a real file system [12]. The cryptographic file system can be also as a low level encryption layer operates at the lower level of abstraction under the real file system to encrypt the entire disk or partition [13].

III. PROBLEMS AND CHALLENGES OF STORAGE ENCRYPTION

Designing a cryptosystem for storage domain is difficult and error-prone task that may affect the system security and performance, if it is not properly implemented [14]. This is due to a number of issues for this state of affairs that should be considered while developing a practical storage cryptosystem. The first issue is related to the long latency of stored data which is in possibly years, where the storage latency roughly indicates the amount of time that the attacker needs to analyze the security applied. Unlike ephemeral transmission where the security still occurs within a short period of time and both of the communicators can participate in various protection methods to ensure a secure communication, such as cryptographic methods, key agreement, authentication protocols, etc. [9].

Cryptography is known to be a mathematically heavy operation, and the disk storage is manipulating with quantum amount of stored data that need to be encrypted or decrypted each time. Therefore, the frequent read and write of stored files with cryptographic service, as well as including primary operations for management, operational and technical controls that involved during data encryption will have a greater impact on the system performance and responding time, which thus resulted in the reduction of a system usage [15].

As a matter of fact, the selection of in-appropriate cryptographic methods and related variables leads the system to be easily broken even without a user is being aware of this fact. However, this can be sometimes more dangerous than using no cryptography at all, since the user can be deceiving because of his thought the data remains confidential in the storage [16].

Key management is another challenging issue, where the risk occurs when reading encrypted data with a lost or forgetting keys. Therefore, any problem occurs with any of the keys can render the data inaccessible when it is requested, and creating any new unrelated key to the previously encrypted data will do not help. On the other hand, storing keys in plain on disk or sharing them among different users is not a desired option in storage domain [17].

Backup of storage data is another attractive issue to the confidential storage when a user is not using a proper a secure backup method with the encryption scheme. Here, the attacker can simply collect the backup versions and then extracts the plaintexts without even access the keys. The in-place update of file contents to the same file location may lead to compromise the uniqueness of keys and IVs relative to data content if the keys and IVs are generated as a function of data positions within a file or storage medium [10].

IV. SECURITY REQUIREMENTS FOR STORAGE CRYPTOSYSTEMS

A storage cryptosystem for electronic documents protection should be able to ensure of documents security, authentication, confidentiality, and maintain privacy and integrity. Also, it should be able to guarantee of the high read and write response efficiency. In this section, we define the security requirements of storage encryption for electronic documents.

Security: a cryptosystem must grant a high amount of security attached within cryptography for protecting digital documents. Typically, selecting the appropriate encryption algorithm and related parameters, the operation mode, the length and kind of the secret keys, and using a combination of symmetric and asymmetric algorithms, are the important factors in a cryptographic system to provide maximum security strength [10]. The stronger cryptographic scheme should be also able to control and manage the usage of keys, prevention and detection of any illegal or unauthorized action to reveal the keys [18]. However, the effective way to prove security level of any cryptosystem is to rigorously prove the analyticity of the applied cryptographic scheme, where the stronger security is the stronger for resisting any kind of cryptanalysis attack [14].

Integrity: the property where the documents hold when they have not unauthorized modified. In the documents encryption systems, the integrity means that the documents are kept in storage domain as they are supposed to be without tampering. This can be provided using several techniques, such as digital signature, one-way hash functions, Message Authentication Code (MAC) and the combined approach (HMAC) from the one-way hash and the MAC, to provide higher integrity level.

Authentication: the process of verifying a principles claimed identity. The authentication in the storage cryptosystems is provided by the logon-password. However, keeping users' list passwords in a machine will become a target for attacker. Moreover, encrypting the password also will not help, because the user needs to decrypt the password each time

logon the system, which hence may be compromised using dictionary attack. Various proposed mechanisms can be followed to improve a password protection, such as salting password, time-stamp and one-time password, etc.

Confidentiality: also known as secrecy or privacy, where the breaches of confidentiality range from embarrassing to the disastrous. It guarantees that storage document files cannot be accessed by unauthorized parties. This can be realized by using cryptography in addition to the access control mechanisms through keeping documents stored in encrypted form only, and the system must contain the right access control to decrypt them [1].

High response efficiency: the encryption performance in the software-based varies between using user space encryption applications and the cryptographic file systems of different locations. However, the latter approach is usually optimizing the performance because it is getting the encryption and integrity protection techniques together, and all related computations are performed in highly seamless and compatible manner without many data copies between a kernel and a user space [12].

V. PROPOSED MODEL

The overall objective of this work is to address the cryptographic requirements of storage domain by reusing them in more specific and efficient way to design our cryptographic scheme for storage electronic documents. The designed model cans trade-off between security and performance to provide encryption and decryption of stored documents in an automatic and transparent manner, and with minimum interaction from a user. In order to achieve this objective, the encryption, decryption and key management processes should be done transparently, since the minimal amount of user interaction to setup and use the system effectively increases the security and usability [19]. Therefore, a new cryptographic file system filter driver exclusive for stored document files is inserted in a middleware level inside the kernel. This cryptographic filter driver is attached between the I\O manager and file system driver inside the Microsoft Windows kernel.

Further, to emphasize on the following functions: Firstly, the cryptographic filter driver will allow the user to carry on with his work without adding overhead for ciphering or effecting on his normal operations. Secondly, the cryptographic filter driver automatically recognizes the electronic document file once it detects that the process is trying to open or save a document file on the local disk. Thirdly, the cryptographic filter driver automatically encrypts or decrypts the document file contents as per-file basis with high secret and seamless manner. Key management is another task for this cryptographic filter driver since the user will never manipulate with a manual key management and will not pay more attention to the related problems. Figure 1 shows the structure of the file system filter driver and the interaction between other parts inside the Windows kernel.

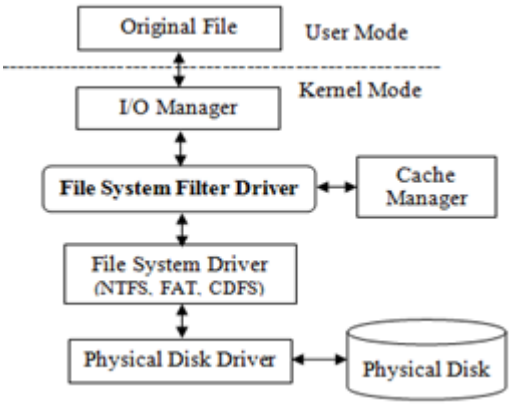

Fig. 1. File system filter driver structure.

A. Working Structure

When a user threads a request to create, read or write a file stored on a local disk, the request will be passed to the I/O Manager layer which sits above the real file system. Here, the I/O Manager makes the required process like parsing the filename, finding the physical location on the local disk, and further builds the I/O Request Packets (IRPs). Finally, it routs the IRPs directly to the appropriate device driver to process the request or a part of the request that it can handle. Any implemented file system filter driver of specified functionality can be attached and integrated between the I/O Manager and the real file system driver. So, when the IRP request is sent to the local disk driver with a specified function call, the attached filter driver will effectively intercept that request to perform its task for which it was being designed.

In accordance, a new cryptographic filter driver is designed to provide a mandatory and transparent encryption, decryption and key management operations for the storage document files. Thus, the cryptographic services would be performed online without any more interaction from the user or changing on his habits. The cryptographic filter driver provides a uniform interface for all applications and underlying kernel file systems. This means the transparent encryption and decryption of a document file will be carried out without being bothered from which application that a document is coming from. In consequence, the designed cryptographic model will be compatible with all applications that work with electronic documents, also provide a user with a very convenient environment to work.

When a request of writing a new document file into a local disk is received, the cryptographic filter driver will automatically encrypt the document contained in the IRP before stored on the local disk. On the other hand, if the IRP request is to read a stored document file, the cryptographic filter driver will transparently decrypt it on-the-fly and then send it in plain form to the user or application issuing a system call. The cryptographic file system filter driver can be easily implemented and deployed to accomplish its features in high efficient and reliable way, and without any required modification on the operating system functions or any of the internal kernel structure.

Figure 2 shows the architecture of a designed model for the cryptographic file system filter driver. Inside the designed model there are two components, the key management unit and the cryptographic unit.

B. Cryptographic Unit

When a user threads a request to write or read an electronic document file, the cryptographic filter driver will immediately initialize the encryption algorithm and other encryption parameters which are realized on the cryptographic unit. The whole document content will be firstly divided into number of blocks of a fixed size of 16 bytes each.

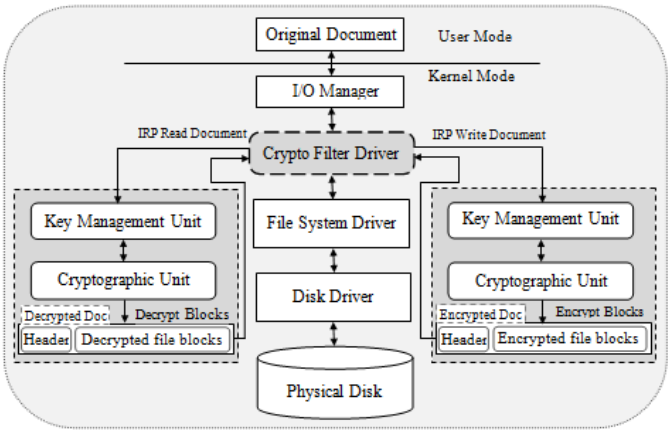

Fig. 2. Design Model of Cryptographic Filter Driver.

The AES as a fast symmetric encryption algorithm with 128-bit default key length is picked. We also chose the Cipher Block Chaining (CBC) as an encryption mode. In order to guarantee of achieving a better security level, the uniqueness requirement for the initialization vector (IV) across all document files which are encrypted under a given encryption key should be realized.

Similarly, read or copy a stored document file is performed transparently in reverse order. When a cryptographic filter driver recognizes that the IRP request is to read a stored document from a local disk. The cryptographic filter driver immediately responds once the read operation is completing from the disk, and subsequently performs the decryption process. The key management unit is firstly recalling the encryption key and the corresponding IV. Subsequently, the cryptographic unit will initiate the encryption cipher and related parameters that are required for decryption operation. It then uses the encryption key to decipher all document blocks. Once the plain document is generated, it will be directly sent back to the caller on the user space.

C. Key Management Unit

Losing or forgetting encryption keys due to the long-time storage means losing access to all corresponding encrypted data stored on disk. Moreover, storing keys in plain form on the hard disk will increase its chances to be stolen and leaked easily. Therefore, in order to enforce the security of the encryption keys, and to reduce the risks come from brute-force attack, the cryptographic filter driver will be used to protect, manage and monitor the keys.

Key management unit involves the operations of creating, using and retaining the encryption keys. However, using a single key to encrypt all document files is not secure, since if the attacker successes to obtain the secret key for one document file, he would be able to recover all other encrypted documents.

Therefore, in this design model, each document file will be encrypted using different symmetric encryption key generated randomly. In order to provide a secure protection for the used symmetric encryption keys, we proposed to use a public key encryption algorithm to encrypt the file's encryption key using a corresponding user's public key. The encrypted key will be then stored as extended attributes on the header of the document file. So, once the encrypted document file is being read from the local disk, the key management unit directly extracts the encrypted key from the header file and calls the user's private key to decrypt the symmetric key. Finally, it sends the encryption key to the cryptographic unit to perform the decryption operation.

VI. CONCLUSION

In this paper, we have described the challenges and issues related cryptography on the storage domain. Then we studied the cryptographic requirements that should be considered while designing a cryptographic system for storage domain. Therefore, we based on the reuse approach for the identified cryptographic requirements to design a transparent storage encryption model that uses a file system filter driver technology for Windows. The model is able to dynamically and transparently encrypt and decrypt the electronic document files stored on the local disk, on the fly. The cryptographic model structured from two component units are the cryptographic and key management units. This paper also detailed analyzes the key technologies which can be used to implement the proposed model.

REFERENCES

[1] S. Na, and S. Lee, "Design of security mechanism for electronic document repository system," IEEE International Conference on Convergence and Hybrid Information Technology, pp. 708-715, 2008.

[2] A. Lamsweerde, S. Brohez, R. Landtsheer and D. Janssens, "From system goals to intruder anti-goals: attack generation and resolution for security requirements engineering," Proceedings of the RE'03 Workshop on Requirements for High Assurance Systems, Monterey (CA), pp. 49-56, 2003.

[3] D. Firesmith, "Specifying reusable security requirements," Journal of Object Technology, vol.3, pp. 61-75, 2004.

[4] R. Villarroel, E. Ferna´ndez-Medina and M. Piattini, "Secure information systems development- a survey and comparison," Journal of Computer & Security, vol.24, pp. 308-321, 2005.

[5] G. Sindre, D. Firesmith and A. Opdahl, "A reuse-based approach to determining security requirements," the 9th International Workshop on Requirements Engineering: Foundation for Software Quality, pp. 16-25, 2003.

[6] J. Jensen, I. Tøndel and H. Andresen, "Reusable security requirements for healthcare applications," IEEE International Conference on Availability, Reliability and Security, pp. 380-385, 2009.

[7] D. Mellado, C. Blanco, L. Sánchez, and E. Fernández-Medina, "A systematic review of security requirements engineering," Journal of Computer Standards & Interfaces, vol.32, pp.153-165, 2010.

[8] S. Konrad, B. Cheng, L. Campbell and R. Wassermann, "Using security patterns to model and analyze security requirements," IEEE Workshop on Requirements for High Assurance Systems, pp. 13-22, 2003.

[9] J. Hughes, "IEEE standard for encrypted storage," Computer, vol.37, pp. 110-112, 2004.

[10] S. Diesburg, C. Meyers, D. Lary and A. Wang, "When cryptography meets storage," 4th ACM international workshop on Storage security and survivability, pp. 11-20, 2008.

[11] M. Szeredi, "Filesystem in Userspace," Available in http://sourceforge.net, 2014.

[12] E. Zadok, I. Badulescu and A. Shender, "Cryptfs: a stackable vnode level encryp-tion file system," Technical Report, No. CUCS-021-98, Computer Science Department, Columbia University, 1998.

[13] R. C. Dowdeswell and J. Ioannidis, "The CryptoGraphic disk driver," the Annual USENIX Technical Conference, pp. 179-186, 2003.
[14] I. Damgard, "A "proof-reading" of some issues in cryptography," the 34th International Colloquium in Automata, Languages and Programming, Springer LNCS, vol.4596, pp. 2-11, 2007.
[15] M. Alomari, K. Samsudin and A. Ramli, "A parallel XTS encryption mode of operation," IEEE Student Conference on Research and Development, pp. 172-175, 2009.
[16] E. R. Weippl, Security in e-learning, Vienna, Austria: Springer, 2005, pp. 131-153.
[17] S. Diesburg and A. Wang, "A survey of confidential data storage and deletion methods". Journal of ACM Computing Surveys, vol.43, 2010.
[18] M.W. Storer, K.M. Greenan and E.L. Miller, "POTSHARDS: Secure long-term storage without encryption," the USENIX Annual Technical Conference, pp. 143–156, 2007.
[19] R.K. Pal and I. Sengupta, "Enhancing file data security in Linux operating system by integrating secure file system," IEEE Symposium on Computational Intelligence in Cyber Security, pp. 45-52, 2009.